REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	5/4/2005	Final Progress Report 11/21/00 - 11/20/04

4. TITLE AND SUBTITLE
Integrated Environment for Control Software Engineering

5. FUNDING NUMBERS
DAAD190110003

6. AUTHOR(S)
Scott A. Smolka, Eugene Stark, Rance Cleaveland

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Department of Computer Science
SUNY Stony Brook
Stony Brook, NY 11794-4400

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U. S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
40026.1-C1

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Significant scientific progress has been made during the final year of the grant. We have continued the development of PIOAL, the process-algebraic specification language for Probabilistic I/O Automata that forms the basis for our tool integration effort. We have also developed a Monte Carlo model checking algorithm a Hybrid-automaton model of cardiac; and a safety-liveness semantics for UML 2.0 Sequence Diagrams. We have moreover pursued the development of mathematical formalisms for the combined modeling of functional and performance aspects of systems, and for software architecture specification.

14. SUBJECT TERMS
Probabilistic Input/Output Automata, Concurrency Workbench, Monte Carlo Model Checking

15. NUMBER OF PAGES
5

16. PRICE CODE

REPORT DOCUMENTATION PAGE (SF298)
(Continuation Sheet)

1 List of Papers Submitted or Published Under ARO Sponsorship

1. E. Stark, "Formally Specifying CARA in Java," *International Journal on Software Tools for Technology Transfer*, Vol. 5, No. 4, pp. 331-350 (2004).

2. E. Stark and W. Song, "Linear Decision Diagrams." Unpublished technical report, available at `http://bsd7.starkhome.cs.sunysb.edu/~stark/REPORTS/ldd.pdf`

3. R. Grosu and S.A. Smolka. "Safety-Liveness Semantics for UML 2.0." *Proceedings of ACSD 2005: Fifth International Conference on Application of Concurrency to System Design*, IEEE Computer Society Press, Los Alamitos, CA, USA, (June 2005).

4. P. Ye, E. Entcheva, R. Grosu, and S.A. Smolka. "Efficient Modeling of Excitable Cells Using Hybrid Automata." *Proceedings of Computational Methods in Systems Biology*, Lecture Notes in Computer Science, Springer-Verlag (April 2005).

5. R. Grosu and S.A. Smolka. "Monte Carlo Model Checking." *Proceedings of TACAS 2005: Eleventh International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science, Springer-Verlag (April 2005).

6. C.W. Keller, D. Saha, S. Basu, and S.A. Smolka. "FocusCheck: A Tool for Model Checking and Debugging Sequential C Programs." *Proceedings of TACAS 2005: Eleventh International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science, Springer-Verlag (April 2005).

7. P. Yang, Y. Dong, C.R. Ramakrishnan, and S.A. Smolka. "Compiling Mobile Processes for Efficient Model Checking" (Winner of Most Practical Paper Award). *Proceedings of Seventh International Symposium on Practical Aspects of Declarative Languages (PADL 05)*, Lecture Notes in Computer Science, Springer-Verlag (Jan. 2005).

8. S. Basu, D. Saha, and S.A. Smolka. "Localizing Program Errors for Cimple Debugging." *Proceedings of 24th IFIP International Conference on Formal Techniques for Networked and Distributed Systems (FORTE 2004)*, Lecture Notes in Computer Science, Springer-Verlag (Sep. 2004).

9. A. Ray and R. Cleaveland. "Unit Verification: the CARA Experience." *Software Tools for Technology Transfer*, 5(4):351-369, May 2004.

10. D. Zhang and R. Cleaveland. "Fast On-the-Fly Model Checking for Data-Based Systems." Submitted for publication.

11. D. Zhang and R. Cleaveland. "Faster Parametric Model Checking for Real Time." Submitted for publication.

12. D. Zhang and R. Cleaveland. "Temporal-Logic Query Checking for Real-Time Systems." Submitted for publication.

2 Scientific Personnel Supported by the Project

Senior Personnel Rance Cleaveland, Scott Smolka, Eugene Stark

Graduate Students Arnab Ray, Wenxin Song, Zan Sun, DeZhuang Zhang, and Wenkai Tan

3 Report of Inventions

4 Scientific Progress and Accomplishments

Significant scientific progress has been made during the fourth and final year of grant DAAD190110019. A main thrust of the fourth year's activities has been aimed at improving the performance of the PIOATool and comparing its performance to the PRISM model checker. We have also designed and implemented the Aristotle runtime verification tool suite and applied it to the Linux kernel, as well as the `GMC`^{2} software model checker for `GCC`. We also developed and implemented a generic, on-the-fly technique for checking the correctness of real-time systems. Finally, we developed a Simulink/Stateflow model of the CARA software system.

4.1 Performance Enhancement of PIOATool

A main thrust of our activities over the past year has been to try to obtain a meaningful comparison of the efficiency of the PIOATool analysis algorithms with those of the PRISM system from the University of Birmingham in the UK. We were particularly interested in comparing the performance of our sparse-matrix package implemented using "linear decision diagrams" (LDDs, discussed in last year's progress report) with the performance of PRISM's package which uses multi-terminal binary decision diagrams (MTBDDs) in the form of the "CUDD" BDD library from the University of Colorado at Boulder. To this end, we created a version of the standard "kanban" benchmark that would compile into an identical CTMC model under both PIOATool and under PRISM. We compared the time taken by PRISM to do a reachability analysis of this model with the time taken by PIOATool to do the same thing using the LDD-based matrix implementation. Although we did not expect the Standard ML-based PIOATool to be quite as fast as the C++-based CUDD library used in PRISM, unfortunately we found initially that PIOATool was substantially slower than PRISM, and that the comparison factor got worse as the size of the problem instances increased.

In order to try to get a better understanding of why PRISM was able to achieve so much better performance than PIOATool, we decided to construct a serious MTBDD implementation in Standard ML, adapt our sparse matrix package so that it could use either an LDD or MTBDD representation, and to see how well we could make the MTBDD-based implementation perform. Since the Standard ML of New Jersey runtime system does not provide the ML programmer with much control over its allocation behavior, we needed to develop some innovative approaches to limit the size of the MTBDD node table and operation caches and avoid thrashing. In addition, a major refactoring of the PIOATool code was required in order to make the reachability and other

algorithms as independent as possible of the underlying matrix representation. In the end, we found that we were able to perform reachability analysis in PIOATool using an MTBDD-based matrix representation at a speed roughly four times slower than that of PRISM/CUDD on the same hardware, with the comparison factor roughly independent of the problem instance, as long as sufficient memory was available. As a result of this work, we were able to conclude that the fact that our LDD implementation did not make use of any sort of operation cache was the main reason for the poor performance, and that in order to build an LDD implementation able to compete favorably with MTBDDs we would somehow have to find a way to incorporate operation caches into the former.

We also compared the performance of PRISM's MTBDD-based iterative algorithm for CTMC steady-state analysis with that of a similar procedure implemented in PIOATool. We found a speed relationship similar to that observed in the case of reachability analysis. However, the PRISM group has previously reported that iterative methods for solving large linear systems do not work particularly well when MTBDDs are used to represent the solution vector, due to an apparent lack of patterns in the solution vector that can be exploited by an MTBDD-based representation. We also found this to be the case in our experiments.

As part of the above-described work, we implemented several iterative solution procedures in the context of PIOATool, including Jacobi iteration with an over-relaxation parameter, successive over-relaxation (SOR), and a block Gauss-Seidel variant that exploits structure obtained from reachability analysis to improve the convergence rate. We supplanted our symbolic procedure for performing steady-state analysis in PIOATool with a numeric procedure suitable for iterative use with MTBDD-based matrix implementations. We also devised and implemented a "compaction" algorithm for MTBDD-based matrices, which can efficiently take such a matrix, together with MTBDD-based descriptions for subsets of the row and column indices, and output a smaller matrix that includes only those rows and columns having indices in the specified sets. We use this algorithm for efficiently converting an MTBDD-based matrix having a huge number of rows and columns, most of which are zero, to a traditional sparse matrix implementation having row and column indices that fit within a single machine word.

4.2 Runtime Verification of the Linux Kernel

We have developed Aristotle, a system that employs a combination of runtime monitoring and Monte Carlo approximation techniques for the development of high-confidence OS kernels. Aristotle includes a kernel-specific compiler (KGCC) that instruments OS kernels with inline monitoring code for checking system-correctness properties at runtime. As an OS executes, instrumented code segments are monitored for compliance with desired properties. Using a runtime adaptation of a novel Monte-Carlo model-checking algorithm we recently developed, Aristotle estimates the confidence that a certain property is correct. When the confidence exceeds a certain threshold, inline monitoring is turned off, therefore regaining the performance lost due to instrumentation overhead.

Aristotle's main benefit is its use of an *adaptive* approach to reduce monitoring overhead. Frequently executed code paths are responsible for the majority of the monitoring overhead—and are therefore the first to have their monitoring turned off—as the system is used more and confidence in its stability grows. At the same time, monitoring of seldom-executed code paths, which are often the hardest to debug, continues as long as the confidence in them has not reach sufficiently high

levels, and since they do not execute frequently, they contribute little to the overhead.

Our prototype system instruments Linux file-system code. Our benchmarking results show that, although instrumentation overhead can be high initially, the system reaches high confidence within acceptably short periods of time, at which point instrumentation can be turned off and the overhead drops to as little as 11%. Our results also illustrate Aristotle's ability to pinpoint kernel-resident bugs, such as object leaks. A paper on the Aristotle system has been submitted for publication.

4.3 Open-Source Model Checking

GMC^2 is a software model checker we recently developed for GCC, the open-source compiler from the Free Software Foundation (FSF). GCC has evolved from a modest C compiler, to a full-blown, multi-language compiler that can generate code for more than 30 target architectures. During the past year, the Tree-SSA branch of GCC has merged with the main line, resulting in the addition of two new intermediate languages: GENERIC, which provides a common infrastructure for abstract syntax tree analysis and optimization; and GIMPLE three-address code, which provides a common infrastructure for CFG (control flow graph) analysis and optimization.

GMC^2, which is part of the GMC static-analysis and model-checking tool suite for GCC under development at SUNY Stony Brook, can be seen as an extension of *Monte Carlo model checking* to the setting of concurrent, procedural programming languages. Monte Carlo model checking is a newly developed technique that utilizes the theory of geometric random variables, statistical hypothesis testing, and random sampling of lassos in Büchi automata to realize a one-sided error, randomized algorithm for LTL model checking. To handle the function call/return mechanisms inherent in procedural languages such as C/C++, the version of Monte Carlo model checking implemented in GMC^2 is optimized for pushdown-automaton models. Our experimental results demonstrate that this approach yields an efficient and scalable software model checker for GCC.

4.4 Efficient On-the-Fly Checking of Real-Time Systems

In a collection of papers that have recently been submitted for publication, we have developed a generic automated method for checking properties of real-time systems. The method relies on the symbolic construction of proofs that systems obey certain properties; by tweaking the proof rules, we obtain procedures for (1) real-time model checking; (2) parameteric real-time model checking, in which systems have parameters that may be varied, resulting in different system models; (3) real-time temporal-logic query checking, in which the goal is to take a formula with "placeholders" and fill them in with the strongest assertions that still make the formula true for the model.

The resulting algorithms are based on top-down proof search. In contrast with existing real-time model-checking approaches, which rely either on "pure forward" or "pure backward" exploration of a model's behavior, our method uses a combination of the two in a natural manner, as a human performing a proof would (prove and remember lemmas, etc.) To assess our results empirically, we ran them on a number of case studies proposed in the literatue. The algorithms we developed exhibited vastly better performance for scenarios when systems fail to satisfy the properties proposed for them (the most usual outcome, in practice, since models need debugging) while matching the best existing techniques when systems do satisfy formulas.

4.5 Simulink/Stateflow Model of CARA

Last year, we reported that we had constructed an executable specification of the Computer Assisted Resuscitation Algorithm (CARA) using a restricted fragment of Java as a specification language. A paper on this effort will appear in a special issue of the journal *Software Tools for Technology Transfer*. This year, using the Java model of CARA as a basis, we have developed a Simulink/Stateflow model of CARA. Simulink/Stateflow is a hybrid-systems graphical modeling notation developed and marketed by The MathWorks, Inc. It has witnessed widespread use within the automotive, aerospace, and military/defense industries. We are now in the process of using the Reactis tool suite of Reactive Systems, Inc. to automatically generate comprehensive yet compact test data for CARA, which we plan to transfer to researchers at the Water Reade Medical Center.

5 Technology Transfer

Cleaveland and Smolka are co-founders, along with Steve Sims, of Reactive Systems, Inc. (RSI), which makes advanced design tools for control-software engineering. RSI's main product is the Reactis tool suite, a companion product to The MathWorks Model-Based design tools. Reactis allows MathWorks users to automatically generate thorough yet compact test suites for Simulink/Stateflow models. It also allows one to visualize the execution of models on generated tests with a highly sophisticated visual simulation environment. The Company is a member of The MathWork's Connections program, and currently has 25 automotive and aerospace customers spread across seven countries. Cleaveland also made over 40 presentations about Reactis to different customers during the year. Part of the technology underpinning Reactis has been influenced by ARO-supported research of Cleaveland and Smolka. To learn more about Reactive Systems, please visit the company web site at `www.reactive-systems.com` or contact Cleaveland or Smolka directly.

In other technology-transfer efforts, Scott Smolka gave a presentation on Monte Carlo model checking at the ARO-sponsored 2004 HCES workshop on High-Confidence Embedded Systems. Cleaveland gave presentations on his and Smolka's experiences in starting Reactis at the 2004 Monterey Workshop in Baden, Austria, and he delivered and invited address on software V&V at the MATLAB EXPO, the premiere model-based software development meeting in Tokyo.